S-storage operators

Karim Nour

$\underline{S}$-STORAGE OPERATORS

Karim NOUR [1]
LAMA - Equipe de Logique, Université de Savoie - 73376 Le Bourget du Lac cedex [2]

Abstract In 1990, J.L. Krivine introduced the notion of storage operator to simulate, for Church integers, the "call by value" in a context of a "call by name" strategy. In this present paper, we define, for every λ-term $\underline{S}$ which realizes the successor function on Church integers, the notion of $\underline{S}$-storage operator. We prove that every storage operator is a $\underline{S}$-storage operator. But the converse is not always true.

Mathematics Subject Classification : 03B40, 68Q60
Keywords : Church integer ; Storage operator ; Call by value ; Call by name ; Head reduction ; Solvable ; Successor ; $\underline{S}$-storage operator.

1 Definitions and notations

- We denote by Λ the set of λ-terms modulo α-equivalence, and by $\mathcal{V}$ the set of λ-variables.
- Let $t, u, u_1, ..., u_n$ be λ-terms, the application of t to u is denoted by $(t)u$. In the same way we write $(t)u_1...u_n$ instead of $(...((t)u_1)...)u_n$.
- The sequence of λ-terms $u_1, ..., u_n$ is denoted $\overline{u}$.
- If $\overline{u} = u_1, ..., u_n$, we denote by $(t)\overline{u}$ the λ-term $(t)u_1...u_n$.
- The β-equivalence relation is denoted by $u \simeq_\beta v$.
- The notation $\sigma(t)$ represents the result of the simultaneous substitution σ to the free variables and the constants of t after a suitable renaming of the bounded variables of t.
- Let us recall that a λ-term t either has a head redex [i.e. $t = \lambda x_1...\lambda x_n(\lambda xu)v\overline{w}$, the head redex being $(\lambda xu)v$], or is in head normal form [i.e. $t = \lambda x_1...\lambda x_n(x)\overline{w}$].
- The notation $u \succ v$ means that v is obtained from u by some head reductions.
- If $u \succ v$, we denote by $h(u, v)$ the length of the head reduction between u and v.
- A λ-term t is said solvable iff the head reduction of t terminates.

The following results are well known (see [3]):

[1] We wish to thank René David for helpful discussions.
[2] e-mail nour@univ-savoie.fr

–If u is β-equivalent to a solvable λ-term, then t is solvable.

–If $u \succ v$, then, for any substitution σ, $\sigma(u) \succ \sigma(v)$, and $h(\sigma(u),\sigma(v))$=h(u,v). In particular, if for some substitution σ, $\sigma(t)$ is solvable, then t is solvable.

- We define $(u)^n v$ by induction : $(u)^0 v = v$ and $(u)^{n+1} v = (u)(u)^n v$.
- For each integer n, we define the Church integer $\underline{n} = \lambda f \lambda x (f)^n x$.
- A closed λ-term $\underline{S}$ is called successor iff, for every $k \geq 0$, $(\underline{S})\underline{k} \simeq_\beta \underline{k+1}$.

Examples Let $\underline{S_1} = \lambda n \lambda f \lambda x (f)((n)f)x$ and $\underline{S_2} = \lambda n \lambda f \lambda x ((n)f)(f)x$.
It is easy to check that $\underline{S_1}$ and $\underline{S_2}$ are successors. □

2 Introduction

In λ-calculus the left reduction strategy (iteration of the head reduction) has much advantages : it always terminates when applied to a normalizable λ-term and it seems more economic since we compute a λ-term only when we need it. But the major drawback of this strategy is that a function must compute its argument every time it uses it. In 1990 J-L. Krivine introduced the notion of storage operators in order to avoid this problem and to simulate call-by-value when necessary.

Let F be a λ-term (a function), and $\underline{n}$ a Church integer. During the computation, by left reduction, of $(F)\theta_n$ (where $\theta_n \simeq_\beta \underline{n}$), θ_n may be computed several times (as many times as F uses it). We would like to transform $(F)\theta_n$ to $(F)\tau_n$ where τ_n is a fixed closed λ-term β-equivalent to $\underline{n}$. We also want this transformation depends only on θ_n (and not F).

Therefore the definition : A closed λ-term T is called storage operator if and only if for every $n \geq 0$, there is a closed λ-term $\tau_n \simeq_\beta \underline{n}$ such that for every $\theta_n \simeq_\beta \underline{n}$, $(T)\theta_n f \succ (f)\tau_n$ (where f is a new variable).

Let's analyse the head reduction $(T)\theta_n f \succ (f)\tau_n$, by replacing each λ-term which comes from θ_n by a new variable.

If $\theta_n \simeq_\beta \underline{n}$, then $\theta_n \succ \lambda g \lambda x (g) t_{n-1}$, $t_{n-k} \succ (g) t_{n-k-1}$ $1 \leq k \leq n-1$, $t_0 \succ x$, and $t_k \simeq_\beta (g)^k x$ $0 \leq k \leq n-1$.

Let x_n be a new variable (x_n represents θ_n). $(T)x_n f$ is solvable, and its head normal form does not begin by λ, therefore it is a variable applied to some arguments. The free variables of $(T)x_n f$ are x_n and f, we then have two possibilities for its head normal form : $(f)\delta_n$ (in this case we stop) or $(x_n)a_1 ... a_m$.

Assume we obtain $(x_n)a_1...a_m$. The variable x_n represents θ_n, and $\theta_n \succ \lambda g\lambda x(g)t_{n-1}$, therefore $(\theta_n)a_1...a_m$ and $((a_1)t_{n-1}[a_1/x, a_2/g])a_3...a_m$ have the same head normal form. The λ-term $t_{n-1}[a_1/g, a_2/x]$ comes from θ_n. Let x_{n-1,a_1,a_2} be a new variable (x_{n-1,a_1,a_2} represents $t_{n-1}[a_1/g, a_2/x]$). The λ-term $((a_1)x_{n-1,a_1,a_2})a_3...a_m$ is solvable, and its head normal form does not begin by λ, therefore it is a variable applied to some arguments. The free variables of $((a_1)x_{n-1,a_1,a_2})a_3...a_m$ are among x_{n-1,a_1,a_2}, x_n, and f, we then have three possibilities for its head normal form : $(f)\delta_n$ (in this case we stop) or $(x_n)b_1...b_r$ or $(x_{n-1,a_1,a_2})b_1...b_r$.

Assume we obtain $(x_{n-1,a_1,a_2})b_1...b_r$. The variable ($x_{n-1,a_1,a_2}$ represents $t_{n-1}[a_1/g, a_2/x]$, and $t_{n-1} \succ (g)t_{n-2}$, therefore $(t_{n-1}[a_1/g, a_2/x])b_1...b_r$ and $((a_1)t_{n-2}[a_1/g, a_2/x])b_1...b_r$ have the same head normal form. The λ-term $t_{n-2}[a_1/g, a_2/x]$ comes from θ_n. Let x_{n-2,a_1,a_2} be a new variable (x_{n-2,a_1,a_2} represents $t_{n-2}[a_1/g, a_2/x]$). The λ-term $((a_1)x_{n-2,a_1,a_2})b_1...b_r$ is solvable, and its head normal form does not begin by λ, therefore it is a variable applied to arguments. The free variables of $((a_1)x_{n-2,a_1,a_2})b_1...b_r$ are among x_{n-2,a_1,a_2}, x_{n-1,a_1,a_2}, v_n, and f, therefore we have four possibilities for its head normal form : $(f)\delta_n$ (in this case we stop) or $(x_n)c_1...c_s$ or $(x_{n-1,a_1,a_2})c_1...c_s$ or $(x_{n-2,a_1,a_2})c_1...c_s$ ... and so on...

Assume we obtain $(x_{0,d_1,d_2})e_1...e_k$ during the construction. The variable x_{0,d_1,d_2} represents $t_0[d_1/g, d_2/x]$, and $t_0 \succ x$, therefore $(t_0[d_1/g, d_2/x])e_1...e_k$ and $(d_2)e_1...e_k$ have the same head normal form ; we then follow the construction with the λ-term $(d_2)e_1...e_k$. The λ-term $(T)\theta_n f$ is solvable, and has $(f)\tau_n$ as head normal form, so this construction always stops on $(f)\delta_n$. We can prove by a simple argument that $\delta_n \simeq_\beta \underline{n}$.

According to the previous construction, the reduction $(T)\theta_n f \succ (f)\tau_n$ can be divided into two parts :

- A reduction that does not depend on n :

$$\begin{aligned}
(T)x_n f &\succ (x_n)a_1...a_m \\
((a_1)x_{n-1,a_1,a_2})a_3...a_m &\succ (x_{n-1,a_1,a_2})b_1...b_r \\
((a_1)x_{n-2,a_1,a_2})b_1...b_r &\succ (x_{n-2,a_1,a_2})b_1...b_r \\
&\vdots
\end{aligned}$$

- A transformation that depends on n (and not on θ_n) :

$$\begin{aligned}
(x_n)a_1...a_m &\rightsquigarrow ((a_1)x_{n-1,a_1,a_2})a_3...a_m \\
(x_{n-1,a_1,a_2})b_1...b_r &\rightsquigarrow ((a_1)x_{n-2,a_1,a_2})c_1...c_s \\
&\vdots
\end{aligned}$$

.

$$(x_{0,d_1,d_2})e_1...e_k \quad \rightsquigarrow \quad (d_1)e_1..e_k$$

We add new constants x_i and $x_{i,a,b,\overline{c}}$ in λ-calculus, and we consider the following set of head reduction rules :

$$\begin{array}{rcl}
(\lambda xu)v\overline{w} & \succ & (u[v/x])\overline{w} \\
(x_{i+1})ab\overline{c} & \succ & ((a)x_{i,a,b,\overline{c}})\overline{c} \\
(x_0)ab\overline{c} & \succ & (b)\overline{c} \\
(x_{i+1,a,b,\overline{c}})\overline{w} & \succ & ((a)x_{i,a,b,\overline{w}})\overline{w} \\
(x_{0,a,b,\overline{c}})\overline{w} & \succ & (b)\overline{w}
\end{array}$$

We write $t \succ_x t'$ if t' is obtained from t by applying these rules finitely many times.

With this formalisme we have the following result (see [1] and [4]):
A closed λ-term T is a storage operator iff for every $n \geq 0$, $(T)x_nf \succ_x (f)\tau_n$ and where τ_n is a closed λ-term β-equivalent to $\underline{n}$.

The constants x_i and $x_{i,a,b,\overline{c}}$ represent intuitively the λ-terms which come from a non calculated Church integer. The uniform shape of Church integers allows to describe the behaviour of these constants when they are in the head position. However, another method to describe a Church integer is simply to say that it is zero or a successor.

Formally, we add new constants X_i et $X_{i,a,b,\overline{c}}$ in λ-calculus, and we consider, for every successor $\underline{S}$, the following set of head reduction rules :

$$\begin{array}{rcl}
(\lambda xu)v\overline{w} & \succ & (u[v/x])\overline{w} \\
(X_{i+1})ab\overline{c} & \succ & ((\underline{S})X_{i,a,b,\overline{c}})ab\overline{c} \\
(X_0)ab\overline{c} & \succ & (\underline{0})ab\overline{c} \\
(X_{i+1,a,b,\overline{c}})uv\overline{w} & \succ & ((\underline{S})X_{i,u,v,\overline{w}})uv\overline{w} \\
(X_{0,a,b,\overline{c}})uv\overline{w} & \succ & (\underline{0})uv\overline{w}
\end{array}$$

We write $t \succ_X t'$ if t' is obtained from t by applying these rules finitely many times.

A $\underline{S}$-storage operator is defined as follows :
A closed λ-term T is a $\underline{S}$-storage operator iff for every $n \geq 0$, $(T)X_nf \succ_X (f)\tau_n$ where τ_n is a closed λ-term β-equivalent to $\underline{n}$.

This paper studies the link betwen the storage operators and the $\underline{S}$-storage operators. We prove that every storage operator is a $\underline{S}$-storage operator. But the converse is not always true.

3 Storage operators and $\underline{S}$-storage operators

Definition Let T be a closed λ-term. We say that T is a storage operator iff for every $n \geq 0$, there is a closed [3] λ-term $\tau_n \simeq_\beta \underline{n}$, such that for every $\theta_n \simeq_\beta \underline{n}$, $(T)\theta_n f \succ (f)\tau_n$.

Remark Let F be any λ-term (for a function), and θ_n a λ-term β-equivalent to $\underline{n}$. During the computation of $(F)\theta_n$, θ_n may be computed each time it comes in head position. Instead of computing $(F)\theta_n$, let us look at the head reduction of $(T)\theta_n F$. Since it is $\{(T)\theta_n f\}[F/f]$, we shall first reduce $(T)\theta_n f$ to its head normal form, which is $(f)\tau_n$, and then compute $(F)\tau_n$. The computation has been decomposed into two parts, the first being independent of F. This first part is essentially a computation of θ_n, the result being τ_n, which is a kind of normal form of θ_n. So, in the computation of $(T)\theta_n F$, θ_n is computed first, and the result is given to F as an argument, T has stored the result, before giving it, as many times as needed, to any function. □

Examples Let $\underline{S}$ be a successor. If we take :
$T_1 = \lambda n((n)G)\delta$ where $G = \lambda x\lambda y(x)\lambda z(y)(\underline{S})z$ and $\delta = \lambda f(f)\underline{0}$
$T_2 = \lambda n\lambda f(((n)F)f)\underline{0}$ where $F = \lambda x\lambda y(x)(\underline{S})y$,
then it is easy to check that (see [1] and [3]):
for every $\theta_n \simeq_\beta \underline{n}$, $(T_i)\theta_n f \succ (f)(\underline{S})^n\underline{0}$ ($i = 1$ or 2).
Therefore T_1 and T_2 are storage operators. □

Let $\{x_i\}_{i\geq 0}$ be a set of different constants. We define a set of terms (denoted by Λ_x) in the following way :

- If $x \in \mathcal{V} \bigcup \{x_i\}_{i\geq 0}$, then $x \in \Lambda_x$;
- If $x \in \mathcal{V}$, and $u \in \Lambda_x$, then $\lambda x u \in \Lambda_x$;
- If $u \in \Lambda_x$, and $v \in \Lambda_x$, then $(u)v \in \Lambda_x$;
- If $n \in \mathbb{N}$, and $a, b, \overline{c} \in \Lambda_x$, then $x_{n,a,b,\overline{c}} \in \Lambda_x$.

$x_{n,a,b,\overline{c}}$ is considered as a constant which does not appear in $a, b, \overline{c}$.
The terms of the set Λ_x are called λx-terms.

We have the following result (see [1] and [4]) :
A closed λ-term T is a storage operator iff for every $n \geq 0$, there is a finite sequence of head reduction $\{U_i \succ V_i\}_{1\leq i\leq r}$ such that :

1) U_i and V_i are λx-terms ;

[3] In his definition of storage operator, J.L. Krivine autorizes the τ_n to contain free variables which are replaced by terms depend of θ_n. The results of this paper remain valid with this definition but the proofs will be too technical.

2) $U_1 = (T)x_n f$ *and* $V_r = (f)\tau_n$ *where* τ_n *is closed* λ*-term* β*-equivalent to* $\underline{n}$ *;*

3) $V_i = (x_n)ab\overline{c}$ *or* $V_i = (x_{l,a,b,\overline{c}})\overline{d}$ $0 \leq l \leq n-1$*;*

4) If $V_i = (x_n)ab\overline{c}$*, then* $U_{i+1} = (b)\overline{c}$ *if* $n = 0$ *and* $U_{i+1} = ((a)x_{n-1,a,b,\overline{c}})\overline{c}$ *if* $n \neq 0$ *;*

5) If $V_i = (x_{l,a,b,\overline{c}})\overline{d}$ $0 \leq l \leq n-1$*, then* $U_{i+1} = (b)\overline{d}$ *if* $l = 0$ *and* $U_{i+1} = ((a)x_{l-1,a,b,\overline{d}})\overline{d}$ *if* $l \neq 0$*.*

Definitions
1) Let $\{X_i\}_{i\geq 0}$ be a set of different constants. We define a set of terms (denoted by Λ_X) in the following way :

- If $x \in \mathcal{V} \bigcup \{X_i\}_{i\geq 0}$, then $x \in \Lambda_X$;

- If $x \in \mathcal{V}$, and $u \in \Lambda_X$, then $\lambda xu \in \Lambda_X$;

- If $u \in \Lambda_X$, and $v \in \Lambda_X$, then $(u)v \in \Lambda_X$;

- If $n \in \mathbb{N}$, and $a, b, \overline{c} \in \Lambda_X$, then $X_{n,a,b,\overline{c}} \in \Lambda_X$.

$X_{n,a,b,\overline{c}}$ is considered as a constant which does not appear in $a, b, \overline{c}$.
The terms of the set Λ_X are called λX-terms.

2) Let $\underline{S}$ be a successor. A closed λ-term T is called a $\underline{S}$-storage operator iff for every $n \geq 0$, there is a finite sequence of head reduction $\{U_i \succ V_i\}_{1\leq i\leq r}$ such that :

1) U_i and V_i are λX-terms ;

2) $U_1 = (T)X_n f$ and $V_r = (f)\tau_n$ where τ_n is closed λ-term β-equivalent to $\underline{n}$;

3) $V_i = (X_n)ab\overline{c}$ or $V_i = (X_{l,a,b,\overline{c}})uv\overline{w}$ $0 \leq l \leq n-1$;

4) If $V_i = (X_n)ab\overline{c}$, then $U_{i+1} = (\underline{0})ab\overline{c}$ if $n = 0$ and $U_{i+1} = ((\underline{S})X_{n-1,a,b,\overline{c}})ab\overline{c}$ if $n \neq 0$;

5) If $V_i = (X_{l,a,b,\overline{c}})uv\overline{w}$ $0 \leq l \leq n-1$, then $U_{i+1} = (\underline{0})uv\overline{w}$ if $l = 0$ and $U_{i+1} = ((\underline{S})X_{l-1,u,v,\overline{w}})uv\overline{w}$ if $l \neq 0$.

Examples It is easy to check that, for $1 \leq i, j \leq 2$, the above operator T_i is an $\underline{S_j}$-storage operator. We check here (for example) that T_2 is an $\underline{S_2}$-storage operator:
Let n be an integer.
If $n = 0$, then we check that $(T_2)X_n f \succ (X_n)F\ f\ \underline{0}$ and $(\underline{0})\ F\ f\ \underline{0} \succ (f)\underline{0}$.

If $n \neq 0$, then we check that:

$$\begin{array}{rcl}
(T)X_n f & \succ & (X_n)\ F\ f\ \underline{0} \\
((\underline{S_2})X_{n-1,F,f,\underline{0}})\ F\ f\ \underline{0} & \succ & (X_{n-1,F,f,\underline{0}})\ F\ (F)f\ \underline{0} \\
& . & \\
& . & \\
& . & \\
((\underline{S_2})X_{0,F,(F)^{n-1}f,\underline{0}})\ F\ (F)^{n-1}f\ \underline{0} & \succ & (X_{0,F,(F)^{n-1}f,\underline{0}})\ F\ (F)^n f\ \underline{0} \\
(\underline{0})\ F\ (F)^n f\ \underline{0} & \succ & (F)^n f\ \underline{0}
\end{array}$$

We prove (by induction on k) that, for every λ-term u, and for every $0 \leq k \leq n$, we have $(F)^k f\ u \succ (f)(\underline{S_2})^k u$.

- For $k = 0$, it is true.

- Assume that is true for k, and prove it for $k+1$.
$(F)^{k+1} f\ u = (F)(F)^k f\ u \succ (F)^k f\ (\underline{S_2})u$. By induction hypothesis we have that for every λ-term v, $(F)^k f\ v \succ (f)(\underline{S_2})^k v$, then $(F)^{k+1} f\ u \succ (f)(\underline{S_2})(\underline{S_2})^k u = (f)(\underline{S_2})^{k+1} u$.

In particular, for $u = \underline{0}$ and $k = n$, we have $(F)^n f\ \underline{0} \succ (f)(\underline{S_2})^n \underline{0}$.
Therefore T_2 is a $\underline{S_2}$-storage operator. □

A question arizes : **Is there a link between the storage operators and the $\underline{S}$-storage operators ?**

4 Link between the storage operators and the $\underline{S}$-storage operators

Theorem 1 *If T is a storage operator, then, for every successor $\underline{S}$, T is a $\underline{S}$-storage operator.*

Proof Let $\underline{S}$ be a successor and T a storage operator.
Then for every $n \geq 0$, there is a closed λ-term $\tau_n \simeq_\beta \underline{n}$ such that for every $\theta_n \simeq_\beta \underline{n}$, $(T)\theta_n f \succ (f)\tau_n$. In particular $((T)(\underline{S})^n \underline{0})f \succ (f)\tau_n$.
Let $\sigma : \Lambda_X \to \Lambda$ the simultaneous substitution defined by :

$$\begin{array}{c}
\sigma(X_n) = (\underline{S})^n \underline{0} \\
\text{for every } 0 \leq k \leq n-1,\ \sigma(X_{k,a,b,\overline{c}}) = (\underline{S})^k \underline{0} \\
\sigma(x) = x \text{ if } x \neq X_n, X_{k,a,b,\overline{c}}
\end{array}$$

For every $n \geq 0$, we construct a set of head equation $\{U_i \succ V_i\}_{1 \leq i \leq r}$ such that :

1) U_i and V_i are λX-terms ;

2) $V_r = (f)\delta_n$;

3) for every $1 \leq i \leq r-1$, $V_i = (X_n)ab\overline{c}$ or $V_i = (X_{l,a,b,\overline{c}})uv\overline{w}$;

4) $\sigma(V_i)$ is solvable.

Let $U_1 = (T)X_n f$. We have $\sigma(U_1) = ((T)(\underline{S})^n\underline{0})f$ is solvable, then U_1 is solvable and $U_1 \succ V_1$ where $V_1 = (f)\delta_n$ or $V_1 = (X_n)ab\overline{c}$. It is clear that $\sigma(V_1)$ is solvable.
Assume that we have the head reduction $U_k \succ V_k$ and $V_k \neq (f)\delta_n$.

- If $V_k = (X_n)ab\overline{c}$, then, by induction hypothesis, $\sigma(V_k) = ((\underline{S})^n)\underline{0})\sigma(a)\sigma(b)\overline{\sigma(c)}$ is solvable.

  - If $n = 0$, let $U_{k+1} = (\underline{0})ab\overline{c}$. Then $\sigma(U_{k+1}) = (\underline{0})\sigma(b)\sigma(b)\overline{\sigma(c)}$ is solvable.

  - If $n \neq 0$, let $U_{k+1} = (\underline{(S)}X_{n-1,a,b,\overline{c}})ab\overline{c}$. Then $\sigma(U_{k+1}) =$
  $((\underline{S})(\underline{S})^{n-1})\underline{0})\sigma(a)\sigma(b)\overline{\sigma(c)} = \sigma(V_k)$ is solvable.

- If $V_k = (X_{l,a,b,\overline{c}})uv\overline{w}$, then, by induction hypothesis, $\sigma(V_k) = ((\underline{S})^l)\underline{0})\sigma(u)\sigma(v)\overline{\sigma(w)}$ is solvable.

  - If $l = 0$, let $U_{k+1} = (\underline{0})uv\overline{w}$. Then $\sigma(U_{k+1}) = (\underline{0})\sigma(u)\sigma(v)\overline{\sigma(w)}$ is solvable.

  - If $l \neq 0$, let $U_{k+1} = (\underline{(S)}X_{l-1,u,v,\overline{w}})uv\overline{w}$. Then $\sigma(U_{k+1}) =$
  $((\underline{S})(\underline{S})^{l-1})\underline{0})\sigma(u)\sigma(v)\overline{\sigma(w)} = \sigma(V_k)$ is solvable.

Therefore U_{k+1} is solvable and $U_{k+1} \succ V_{k+1}$ where $V_{k+1} = (f)\delta_n$ or $V_{k+1} = (X_n)a'b'\overline{c'}$ or $V_{k+1} = (X_{r,a',b',\overline{c'}})a''b''\overline{c''}$. Since $\sigma(U_{k+1})$ is solvable, then $\sigma(V_{k+1})$ is also solvable.

This constraction always terminates (i.e there is a $r \geq 0$ such that $V_r = (f)\delta_n$). Indeed, if not, we check easily that the λ-term $((T)(\underline{S})^n\underline{0})f$ is not solvable.

Let y be a variable, $\hat{\underline{S}} = (\lambda x\underline{S})y$, and $\hat{\underline{0}} = (\lambda x\underline{0})y$.
Let $\hat{\sigma} : \Lambda_X \to \Lambda$ the simultaneous substitution defined by :

$$\hat{\sigma}(X_n) = (\hat{\underline{S}})^n\hat{\underline{0}}$$
$$\text{for every } 0 \leq k \leq n-1,\ \hat{\sigma}(X_{k,a,b,\overline{c}}) = (\hat{\underline{S}})^k\hat{\underline{0}}$$
$$\hat{\sigma}(x) = x \text{ if } x \neq X_n, X_{k,a,b,\overline{c}}$$

Since $(\hat{\underline{S}})t \succ (\underline{S})t$ and $\hat{\underline{0}} \succ \underline{0}$, we check easily that $((T)(\hat{\underline{S}})^n\hat{\underline{0}})f \succ (f)\hat{\sigma}(\delta_n)$. But $(\hat{\underline{S}})^n\hat{\underline{0}} \simeq_\beta \underline{n}$, then $((T)(\hat{\underline{S}})^n\hat{\underline{0}})f \succ (f)\tau_n$. Therefore $\hat{\sigma}(\delta_n) = \tau_n$. Since τ_n is closed, then δ_n is also closed and $\delta_n = \tau_n \simeq_\beta \underline{n}$.

Therefore T is a $\underline{S}$-storage operator. □

Definition We say that a λX-term U satisfies the property (P) iff for each constant $X_{l,a,b,\overline{c}}$ of U we have :

- $a, b, \overline{c}$ satisfy (P)

- $X_{l,a,b,\overline{c}}$ is applied to a and b ;

- a, b do not contain free variables which are bounded in U.

Lemma 1 *Let U, V be λX-terms which do not begin by λ. If U satisfies (P) and $U \succ V$, then V satisfies (P).*

Proof It is enough to do the proof for one step of head reduction. We have $U = (\lambda x u)v\overline{w}$ and $V = (u[v/x])\overline{w}$. Since U satisfies (P), then $u, v, \overline{w}$ satisfy (P) and x is not free in a, b if the constant $X_{l,a,b,\overline{d}}$ appears in u. Therefore $u[v/x], u_1, ..., u_m$ satisfy (P) and V satisfies (P). □

Let $\triangle : \Lambda_x \to \Lambda_X$ the simultaneous substitution defined by :

$$\triangle(x_n) = X_n$$
$$\text{for every } 0 \leq k \leq n-1,\ \triangle(x_{k,a,b,\overline{c}}) = (X_{k,\triangle(a),\triangle(b),\overline{\triangle(c)}})\triangle(a)\triangle(b)$$
$$\sigma(x) = x \text{ if } x \neq x_n, x_{k,a,b,\overline{c}}$$

Lemma 2 *If U is a λX-term satisfies (P), then there is a λx-term U' such that $\triangle(U') = U$.*

Proof By induction on U.

- For $U = x$, it is true.

- If $U = \lambda x V$, then V satisfies (P), and, by induction hypothesis, there is a λx-term V such that $\triangle(V') = V$. We put $U' = \lambda x V'$. We have $\triangle(U') = U$.

- If $U = (U_1)U_2$ (where U_1 does not begin by a constant), then U_1, U_2 satisfy (P), and, by induction hypothesis, there are λx-terms U_1', U_2' such that $\triangle(U_1') = U_1$ and $\triangle(U_1') = U_1$. We put $U' = (U_1')U_2'$. We have $\triangle(U') = U$.

- If $U = (X_{k,a,b,\overline{c}})ab\overline{V}$, then $a, b, \overline{c}, \overline{V}$ satisfy (P), and, by induction hypothesis, there are λx-terms $a', b', \overline{c'}, \overline{V'}$ such that $\triangle(a') = a$, $\triangle(b') = b$, $\triangle(\overline{c'}) = \overline{c}$, and $\triangle(\overline{V'}) = \overline{V}$. We put $U' = (x_{k,a',b',\overline{c'}})\overline{V'}$. We have $\triangle(U') = U$. □

Theorem 2 *T is a $\underline{S_1}$-storage operator iff T is a storage operator.*

Proof Let $n \geq 0$. If T is a $\underline{S_1}$-storage operator, then there is a finite sequence of head reduction $\{U_i \succ V_i\}_{1 \leq i \leq r}$ such that :

1) U_i and V_i are λX-terms ;

2) $U_1 = (T)X_n f$ and $V_r = (f)\tau_n$ where τ_n is closed λ-term β-equivalent to $\underline{n}$;

3) $V_i = (X_n)ab\overline{c}$ or $V_i = (X_{l,a,b,\overline{c}})uv\overline{w}$ $0 \leq l \leq n-1$;

4) If $V_i = (X_n)ab\overline{c}$, then $U_{i+1} = (\underline{0})ab\overline{c}$ if $n = 0$ and $U_{i+1} = ((\underline{S_1})X_{n-1,a,b,\overline{c}})ab\overline{c}$ if $n \neq 0$;

5) If $V_i = (X_{l,a,b,\overline{c}})uv\overline{w}$ $0 \leq l \leq n-1$, then $U_{i+1} = (\underline{0})uv\overline{w}$ if $l = 0$ and $U_{i+1} = ((\underline{S_1})X_{l-1,u,v,\overline{w}})uv\overline{w}$ if $l \neq 0$.

We prove (by induction on i) that, for every $1 \leq i \leq r$, V_i satisfies (P).

- For $i = 1$, it is true.

- Assume that is true for i, and prove it for $i+1$.

If $V_i = (X_n)ab\overline{c}$, we have two cases :

- if $n = 0$, then $U_{i+1} = (\underline{0})ab\overline{c}$. By induction hypothesis V_i satisfies (P), then $a, b, \overline{c}$ satisfy (P), therefore U_{i+1} and V_{i+1} satisfy (P).
- if $n \neq 0$, then $U_{i+1} = ((\underline{S})X_{n-1,a,b,\overline{c}})ab\overline{c}$. By induction hypothesis V_i satisfies (P), then $a, b, \overline{c}$ satisfy (P). Since $U_{i+1} \succ ((a)(X_{n-1,a,b,\overline{c}})ab)\overline{c}$, then V_{i+1} satisfies (P).

If $V_i = (X_{l,a,b,\overline{c}})uv\overline{w}$ $0 \leq l \leq n-1$, then $u = a$, $v = b$, and $\overline{w} = \overline{c}$ since, by induction hypothesis, V_i satisfies (P). We have two cases :

- if $n = 0$, then $U_{i+1} = (\underline{0})ab\overline{c}$. By induction hypothesis V_i satisfies (P), then $a, b, \overline{c}$ satisfy (P), therefore U_{i+1} and V_{i+1} satisfy (P).
- if $n \neq 0$, then $U_{i+1} = ((\underline{S})X_{l-1,a,b,\overline{c}})ab\overline{c}$. By induction hypothesis V_i satisfies (P), then $a, b, \overline{c}$ satisfy (P). Since $U_{i+1} \succ ((a)(X_{l-1,a,b,\overline{c}})ab)\overline{c}$, then V_{i+1} satisfies (P).

Therefore there is a finite sequence of head reduction $\{M_i \succ N_i\}_{1 \leq i \leq r}$ such that :

1) M_i and N_i are λX-terms ;

2) $M_1 = (T)X_n f$ and $N_r = (f)\tau_n$ where τ_n is closed λ-term β-equivalent to $\underline{n}$;

3) $N_i = (X_n)ab\overline{c}$ or $N_i = (X_{l,a,b,\overline{c}})ab\overline{d}$ $0 \leq l \leq n-1$;

4) If $N_i = (X_n)ab\overline{c}$, then $M_{i+1} = (b)\overline{c}$ if $n = 0$ and $M_{i+1} = ((a)(X_{n-1,a,b,\overline{c}})ab)\overline{c}$ if $n \neq 0$;

5) If $N_i = (X_{l,a,b,\overline{c}})ab\overline{d}$ $0 \leq l \leq n-1$, then $M_{i+1} = (b)\overline{d}$ if $l = 0$ and $M_{i+1} = ((a)(X_{l-1,a,b,\overline{d}})ab)\overline{d}$ if $l \neq 0$.

Since, for every $1 \leq i \leq r$, M_i and N_i satisfy (P), let M_i' and N_i' the λx-terms such that : $\triangle(M_i') = M_i$ and $\triangle(N_i') = N_i$. We have :

1) M_i' and N_i' are λx-terms ;

2) $M'_1 = (T)x_n f$ and $N'_r = (f)\tau'_n$ where τ'_n is closed λ-term β-equivalent to $\underline{n}$;

3) $N'_i = (x_n)a'b'\overline{c'}$ or $N'_i = (x_{l,a',b',\overline{c'}})\overline{d'}$ $0 \leq l \leq n-1$;

4) If $N'_i = (x_n)a'b'\overline{c'}$, then $M'_{i+1} = (b')\overline{c'}$ if $n = 0$ and $M'_{i+1} = ((a')x_{n-1,a',b',\overline{c'}})\overline{c'}$ if $n \neq 0$;

5) If $N'_i = (x_{l,a',b',\overline{c'}})\overline{d'}$ $0 \leq l \leq n-1$, then $M'_{i+1} = (b')\overline{d'}$ if $l = 0$ and $M'_{i+1} = ((a')x_{l-1,a',b',\overline{d'}})\overline{d'}$ if $l \neq 0$.

Therefore T is a storage operator. □

Theorem 3 *There is a $\underline{S_2}$-storage operator which is a no storage operator.*

Proof Let $T = \lambda x(x)\ a\ b\ \underline{0}\ \underline{S}$ where
$a = \lambda x\lambda y\lambda z((x)(z)(x)II\lambda x\underline{0})\lambda x(\underline{S})(z)x$,
$b = \lambda x\lambda y\lambda z(z)x$,
and $\underline{S}$ a successor.

Let n be an integer.
If $n = 0$, then we check that :

$$\begin{array}{rcl}(T)\ X_n\ f & \succ & (X_n)\ a\ b\ \underline{0}\ \underline{S}\ f \\ (\underline{0})\ a\ b\ \underline{0}\ \underline{S}\ f & \succ & (f)\underline{0}\end{array}$$

If $n \neq 0$, then we check that :

$$\begin{array}{rcl}(T)\ X_n\ f & \succ & (X_n)\ a\ b\ \underline{0}\ \underline{S}\ f \\ ((\underline{S_2})X_{n-1,a,b,\underline{0},\underline{S},f})\ a\ b\ \underline{0}\ \underline{S}\ f & \succ & (X_{n-1,a,b,\underline{0},\underline{S},f})\ a\ (a)b\ \underline{0}\ \underline{S}\ f \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ ((\underline{S_2})X_{0,a,(a)^{n-1}b,\underline{0},\underline{S},f})\ a\ (a)^{n-1}b\ \underline{0}\ \underline{S}\ f & \succ & (X_{0,a,(a)^{n-1}b,\underline{0},\underline{S},f})\ a\ (a)^n b\ \underline{0}\ \underline{S}\ f \\ (\underline{0})\ a\ (a)^n b\ \underline{0}\ \underline{S}\ f & \succ & (a)^n b\ \underline{0}\ \underline{S}\ f\end{array}$$

We define two sequences of λ-terms $(P_i)_{0\leq i\leq n}$ and $(Q_i)_{0\leq i\leq n}$ by :

$Q_0 = \underline{S}$, and, for every $0 \leq k \leq n-1$, we put $Q_{k+1} = \lambda x(\underline{S})(Q_k)x$
$P_0 = \underline{0}$, and, for every $0 \leq k \leq n-1$, we put $P_{k+1} = (Q_k)((a)^{n-k-1}b)II\lambda x\underline{0}$

It is easy to check that, for every $1 \leq k \leq n$, $Q_k \simeq_\beta \lambda x(\underline{S})^{k+1}x$.
We prove (by induction on k) that, for every $0 \leq k \leq n$, we have $(a)^n b\ \underline{0}\ \underline{S}\ f \succ (a)^{n-k}b\ P_k\ Q_k\ f$.

- For $k = 0$, it is true.

- Assume that is true for k, and prove it for $k+1$.
$(a)^{n-k}b\ P_k\ Q_k\ f = (a)\ (a)^{n-k-1}b\ P_k\ Q_k\ f \succ$
$((a)^{n-k-1}b)\ (Q_k)((a)^{n-k-1}b)II\lambda x\underline{0}\ \lambda x(\underline{S})(Q_k)x\ f = (a)^{n-k+1}b\ P_{k+1}\ Q_{k+1}\ f.$

In particular, for $k=n$, we have $(a)^n b\ \underline{0}\ \underline{S}\ f \succ (b)\ P_n\ Q_n\ f \succ (f)P_n$.
$P_n = (Q_{n-1})(b)II\lambda x\underline{0} \simeq_\beta (\lambda x(\underline{S})^n x)(b)II\lambda x\underline{0} \simeq_\beta (\underline{S})^n(\lambda x\underline{0})I \simeq_\beta (\underline{S})^n\underline{0} \simeq_\beta \underline{n}$.

Therefore T is a $\underline{S_2}$-storage operator.

We define a sequence of λ-terms $(P'_i)_{0\leq i\leq n}$ by :

$P'_0 = \underline{0}$, and for every $0 \leq k \leq n-1$, we put $P'_{k+1} = (Q_k)(x_{n-k-1,a,b,P'_{n-k},Q_{n-k},f})IIJ$

We check (as before) that :

$$\begin{array}{rcl}
(T)\ x_n\ f & \succ & (x_n)\ a\ b\ P'_0\ Q_0\ f \\
(a)x_{n-1,a,b,P'_0,Q_0,f}\ P'_0\ Q_0\ f & \succ & (x_{n-1,a,b,P'_0,Q_0,f})\ P'_1\ Q_1\ f \\
 & . & \\
 & . & \\
 & . & \\
(a)x_{0,a,b,P'_{n-1},Q_{n-1},f})\ P'_{n-1}\ Q_{n-1}\ f & \succ & (x_{0,a,b,P'_{n-1},Q_{n-1},f})\ P'_n\ Q_n\ f \\
(b)\ P'_n\ Q_n\ f & \succ & (f)P'_n
\end{array}$$

But $P'_n = (Q_{n-1})(x_{0,a,b,P'_{n-1},Q_{n-1},f})II\lambda x\underline{0}$ is not closed.

Note that $P'_n \simeq_\beta (\underline{S})^n(X_{0,a,b,P'_{n-1},Q_{n-1},f})II\lambda x\underline{0} \not\simeq_\beta \underline{n}$. Indeed, if $(\underline{S})^n(X_{0,a,b,P'_{n-1},Q_{n-1},f})II\lambda x\underline{0} \simeq_\beta \underline{n}$, then $(\underline{S})^n(\lambda x_1\lambda x_2\lambda x_3(\underline{S})\underline{0})II\lambda x\underline{0} \simeq_\beta \underline{n}$, therefore $\underline{n+1} \simeq_\beta \underline{n}$. A contradiction.

Therefore T is a no storage operator. □

References

[1] R. David and K. Nour *Storage operators and directed λ-calculus*
Journal of symbolic logic vol. 60, num. 4, pp. 1054-1086, 1995

[2] J.L. Krivine *Lambda-calcul, types et modèles*
Masson, Paris 1990

[3] J.L. Krivine *Opérateurs de mise en mémoire et traduction de Gődel*
Archiv for Mathematical Logic 30, 1990, pp. 241-267

[4] K. Nour *Opérateurs de mise en mémoire en lambda-calcul pur et typé*
Thèse de Doctorat, Université de Chambéry, 1993